{} => Kanli Bracce হল Special Operatore

যেকোন ধরনের Macro বা Consize Representation এর জন্য {} → Kanli Bracce ব্যবহার করা হয়।

Kanli Bracce ব্যবহারের দুইটা কাজ

1. Macro কে resolve করা
2. Alpha Characters কতবার Occurence হবে, সেটা detect করা

a^ দেওয়া মানে => a নাও থাকতে পারে,
a একবার থাকতে পারে.
a একাধিকবার থাকতে পারে।

a{1,7} এর মানে হল a কে 1 থেকে 7 বারের ভিতর detect করবে কেমন।

a{7} এটা লিখা মানে a কে ঠিক ঠিক 7 বার detect করা।

Regular Expression যেই file এ লিখা হয় সেটার extension হল .l.

০। কাইলে চারটা পার্ট থাকে।

%{ Part→1

%} Part→2

%% Part→3

%% Part→4

Part→1. এখানে মূলত header গুলো থাকে। Part→1 এ যা থাকে তা .l এর Compiler Compile করতে পারে না, তাই অপ্রয়োজনীয় Comment এর কথা চিন্তা করে Code (.c) এর শুরুতে নিয়ে যায়।

Part → 2 এটা Optional Part। এখানে মূলত Macro গুলো define করা হয়।

Part → 3 এখানে Regular Expression লিখা হয়। । Compiler এর জন্য এটা হল Main Part.

Part → 4.

⇒ এখানে main function এর কোড লিখবো,

এটা মূলত লিখা হয়,

```
int main();
    yylex();
    return 0;
}
```

yylex() হল Internal Caller or Constructor for flex Compiler.

এটা মূলত Regular Expression থেকে C Code এ Convert করার intermediate কাজগুলো করবে।

চারটা Step এ Regular Expression → C Code এবং executable file → Convert হয়।

Step-1 .l → .c ফাইল
Step-2 .c → .o ফাইল
↳ object file

Step-3 .o → .exe

Step-4 .exe file টা Run করা।

এই চারটা Step এর জন্য Command লিখা হয় →

flex -t test.l > test.c
↳ -t মানে এখন .l ফাইল .c তে Convert করে।

g++ -c -o test.o test.c
.c ফাইল থেকে .o ফাইল বানানো।

g++ -o test test.o -ll

এখন test মানে test একটা executable file
-ll মানে link library

• ./test
এটা মানে test নামের .exe ফাইলকে Run করানো।

* কোন একটা statement যদি একাধিক Regular Expression rule দিয়ে match করে। তাহলে,
Maximum Character যেই regular expression এ match হয় সেটা execute হবে।
* Tie হলে যেইটা শুরুতে থাকবে সেটা execute হবে।

* Regular Expression লিখে যদি ; অথবা {} দেওয়া হয়, তাহলে কোন action হবে না।

* ;REJECT; লিখা হবে পরের statement এ যাবে।
* | দিলে পরের লাইনের action Perform করবে।
শুধু action Perform করবে।

* Input যদি Regular Expression এর সাথে match করে তাহলে তা save করতে
yytext ব্যবহার করা হয়।
yytext => Recently matched input টাকে store করে রাখে।
* yyleng বলা হয় Input এর size কত সেটা বলে দেয়।
yyleng integer return করে।

1. Write Regular Expression for detecting all types of Variable

Ans:

$$[a\text{-}z\ A\text{-}Z\ \_]\ [a\text{-}z\ A\text{-}Z\ 0\text{-}9\ \_]^*$$

2. Write Regular Expression for all regular integer number.

Ans.

$$0 \mid [1\text{-}9]\ [0\text{-}9]^*$$

3. Write Regular Expression for Valid email address

Ans:

$$[a\text{-}z\ A\text{-}Z]\ [a\text{-}z\ A\text{-}Z\ 0\text{-}9]^*\ @\ [a\text{-}z\ A\text{-}Z\ 0\text{-}9]\ (\cdot\ [a\text{-}z A\text{-}Z])^{+}$$

4. Write Regular Expression for Valid Ip address.

Ans.

$$N = [0\text{-}9] + [1\text{-}9]\ [0\text{-}9] + 1\ [0\text{-}9]\ [0\text{-}9] + 2\ [0\text{-}4]\ [0\text{-}9] + 25\ [0\text{-}5]$$

So, Regular Expression : N.N.N.N

5. Write Regular expression for floating Point numbers

Ans:-

~~[0-9]+ | [0-9]+ . [0-9]+~~

$$\backslash.\ [0\text{-}9]^{+} \mid [0\text{-}9]^{+}\ .\ [0\text{-}9]^{+}$$

5. Write Regular Expression for floating Point

Ans:

1. $[0-9]^+ \mid [0-9]^+ . [0-9]^+$

6. Write Regular Expression for detecting mobile Phone number of ~~bangladesh~~ Bangladesh let it is 11 in length

Ans:

~~(+88) [11~~

$01\ [5-9]\{1,1\}\ [0-9]\{8,8\}$

Code→1

```
%{
#include<stdio.h>
%}

%%
[a-z A-Z _] [a-z A-Z 0-9 _]*     {Printf ("Valid");}

%%

int main ()
{
    yylex();
    return 0;

}
```

Valid Variable input দিলে Valid বলবে,
কিন্তু যেটা Valid হবে না সেটার যতটুকু মিলে যাবে ততটুকুর জন্য
Valid বলবে.

Code-2

```
%{
#include<stdio.h>
%}

%%
[a-zA-Z_][a-zA-Z0-9_]*  {Printf("Valid \n");}
[^a-zA-Z][a-zA-Z 0-9]+  {Printf("Not Valid\n");}

%%

int main()
{
    yylex();
    return 0;

}
```

Code-1 এ যেগুলো Valid Variable না সেগুলো handle করা হয় নি.

এখানে

দিয়ে যেগুলো Valid না সেগুলো handle করা হয়েছে।

Code-3

```
%{
    #include<stdio.h>
%}

%%
[a-z A-Z _] [a-z A-Z 0-9_]* | {Printf("Valid\n");}
[^a-z A-Z _] [a-z A-Z 0-9_]+ {Printf("Not Valid\n");}
%%

int main(){
        yylex();
        return 0;
}
```

এটা দেওয়াতে Correct Regular Expression দিলেও এই লাইনে আসবে এবং [^a-z A-Z _] এটা চেক করবে না, [a-z A-z 0-9_]+ এটা শুধু চেক করবে, শুধু {Printf ("Not Valid\n");}